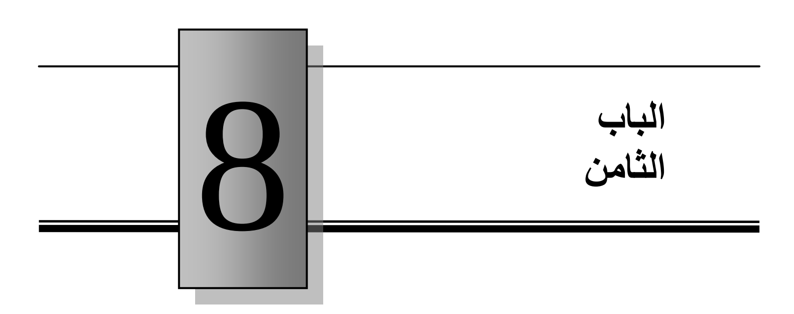
أساسيات وتطبيقات لغة سي

الجزء الخامس

الدكتور عمر زرتي

جامعة طرابلس – ليبيا



المؤشرات

Pointers

8.1 مقدمة

المؤشر pointer هو عبارة عن متغير variable يرمز لموقع في الذاكرة.

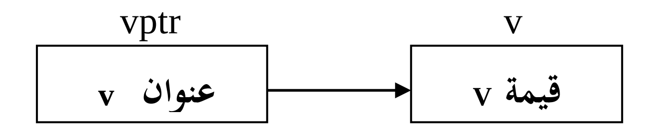
وبما أنه متغير فهوأيضا يحتل موقعا في الذاكرة حيث نضع في هذا الموقع عنوانا لمتغير آخر.

لاحظ أن الذاكرة هي عبارة عن مجموعة من المواقع،كل موقع يسع بايت واحدة، ومن ثم فإن المتغير الصحيح يحتاج إلى موقعين متجاورين، والمتغير الرمزي يحتاج لموقع واحد والمتغير الكسري يحتاج لأربعة مواقع متجاورة.

ويوجد لكل موقع رقم يعبر عن عنوانه. وهذه الأرقام مرتبة تزايديا، بحيث الموقع رقم 100 يليه الموقع رقم 101 ، وهكذا.

فمثلا المتغير v يعبر عن القيمة الموضوعة في الموقع المسمى v ، أما عنوان بداية الموقع فهو متغير آخر نسميه مثلا vptr ، وبالتالى فإن vptr هو عبارة عن مؤشر للمتغيرv .

هذه العلاقة بين المتغير والمؤشر الذي يشير إليه نعبر عنها كما يلي :



لاحظ أن القيمة الموجودة في الخلية vptr هي عنوان الموقع v . في لغة سي نعبر عن هذه العلاقة كمايلي :

vptr = &v

حيث المؤثر & هو مؤثر العنوان address operator . وبطريقة أخرى ، يمكن أن نكتب الجملة :

v = *vptr

وتعنى أن v هي القيمة التي يؤشر لها vptr .

حيث * هنا تسمى مؤثر لا مباشر indirection operator .

لإعلان أن متغير pti هو من النوع المؤشر لنوع صحيح نستخدم الإعلان :

int *pti

ولإعلان أن المتغير ptf مؤشر لنوع عائم ، نستخدم الإعلان

float *ptf

وبالمثل نعلن أن ptd مؤشر لنوع مضاعف بالإعلان التالي :

double *ptd

فإذا كان لدينا مثلا متغير i من النوع الصحيح وكتبنا الجملة

ptri = &i

فإن ptri يتعين له عنوان الموقع i .

وبالمثل فإن الجملة

ptrf = &f

تعين عنوان الموقع f للمؤشر ptrf .

لاحظ ضرورة توافق نوع *p مع vp ، حيث p المؤشر و vp المتغير في الجملة

vp = *p

أو في الجملة

p = &vp

و لطباعة عنوان موقع المتغير v عادة ما نستخدم النظام السادس عشري على النحو التالي :

```
printf( " \n &v = %x " , &v )    ;
```

فمثلاً عند تنفيذ البرنامج التالي :

```
main()
{
        int k, *kp;
        k=18;
        kp=&k;
        printf("\n k=%d",k);
        printf("\n &k=%x",kp);
}
```

الشكل (8.1.1) طباعة عنوان متغير في الذاكرة

نحصل على المخرجات التالية :

k = 18
&k = FDE

حيث FDE هو عنوان k في الذاكرة بالنظام السادس عشري . وبالتالي إذا نظرنا إلى الموقع FDE_{16} من الذاكرة سنجد فيه القيمة 18 وهى قيمة k.

ملاحظة :
لا يجوز أن نعين قيمة عددية من عندنا لأي مؤشر ، ولكن هناك استثناء وحيد لهذه القاعدة وهو قيمة الصفر ، وفى هذه الحالة يسمى بالمؤشر الخالي ، أي الذي لا يؤشر إلى أي شي ، ويرمز له بالاسم NULL .

وجرت العادة أن يتم تعريف NULL بالتوجيه

```
# define   NULL   0
```

بحيث يمكن في البرنامج إجراء الإعلان التالي :

```
float   *pf  =  NULL
```

8.2 المؤشرات والنضائد string pointers

إذا كان s من النوع النضيد ، مثل:

```
char   s[20]
```

فمن الملاحظ عدم إمكانية كتابة جملة مثل

```
s  =  " any  thing "    ;
```

كما هو الحال في بعض اللغات الأخرى ، لأن اسم المصفوفة في لغة سي يعبر عن مؤشر موقع أول عنصر فيها ، وبالتالي سنحصل على رسالة خطأ (Lvalue required) عند تعريف هذه الجملة. يمكننا تصحيح هذا الخطأ بالإعلان :

```
char   s[20]  =  " any  thing "  ;
```

أو نستخدم الدالة strcpy (اختصار string copy) على النحو :

```
strcpy ( s , " any thing " )  ;
```

أو نعين عناصر المصفوفة s[i] الواحد تلو الآخر ، ولكن الأبسط من ذلك كله هو تعريف مؤشر من النوع:

```
char   *sp   ;
```

ثم تعيين النضيد للمؤشر على النحو :

```
sp =  " any  thing "  ;
```

مثـال (8.2.1) : اكتب برنامجا يستخدم المؤشرات لقراءة الاسم وربطه مع النضيد :

" Your name is "

في نضيد واحد .

```
main()
{      char name[12];
        char *stp1, *stp2;
        stp1=&name[0];
        gets(stp1);
        stp2="Your name is ";
        strcat(stp2,stp1);
        puts(stp2);
}
```

الشكل (8.2.1) ربط نضيدين بالدالة strcat

ملاحظات :

1 . يمكن استخدام المؤشرات في متغيرات الدوال

gets , strcat , puts , ...

وبصفة عامة في جميع دوال النضائد .

2 . تعنى الجملة

```
stp1 = &name[0] ;
```

أن المؤشر stp1 هو عنوان أول عنصر في النضيد name . هذه الجملة يمكن أن تكتب أيضاً بالصورة :

```
stp1 = name ;
```

3 . الجملة

```
strcat ( stp2 , stp1 ) ;
```

تعني :

أضف النضيد stp1* إلى النضيد stp2*

4 . الجملة

```
puts ( stp2 ) ;
```

تقوم بطباعة النضيد الذي يشار إلى أول عناصره بالمؤشر stp2 .

<u>مثال</u> (8.2.2) : ماذا يطبع البرنامج التالي ؟

```
main()
{
        char string[9]="pointers";
        char *p;
        int i=0 ,k;
        p=string;
        k=strlen(string);
        printf("\n %d",k);
        while ( i<k )
        {       printf("\n %s",p);
                p++;
                i++;
        }
}
```

الشكل (8.2.2)

سيقوم هذا البرنامج بطباعة الآتي :

```
pointers
ointers
inters
nters
ters
ers
rs
s
```

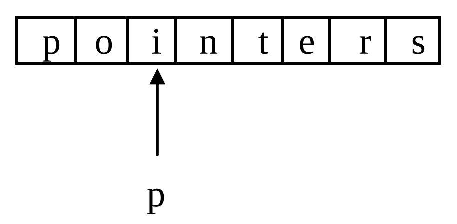
أي أن كلمة pointers ستطبع في البداية ، ثم يحذف منها أول حرف على اليسارفي كل مرة إلى أن يبقى الحرف الأخير فقط من الكلمة.
والسبب في عملية الحذف هذه الجملة

p++ ;

حيث يزداد المؤشر بمقدار واحد فيتحرك إلى اليمين خانة واحدة . وبما أن جملة الطباعة

printf(" \n %s " , p) ;

تطبع ابتداء من الموقع الذي يشير إليه المؤشر p ، فلا تظهر الحروف الواقعة على يسار المؤشر ، فإذا كان p يؤشر مثلاً للحرف i من كلمة pointers كما في الشكل التالي:



فإن جملة الطباعة المذكورة ينتج عنها طباعة الجزء (inters) .

ملاحظات :

1 . الدالة الجاهزة strlen(s) تعطى طول النضيد s ، أي عدد الرموز الذي يحتويها.

2 . المتغير i استخدمناه كعداد لغرض التوقف عند الوصول إلى نهاية النضيد. طبعاً يمكننا استخدام طريقة أخرى وهى اختبار الرمز '0\' الذي يبين نهاية النضيد .

مثال : اكتب برنامجا يقوم بحساب عدد المرات التي يتكرر فيها حرف معين في النضيد :

" pointers are important "

يبين الشكل 8.2.3 البرنامج المطلوب. والغرض من هذا البرنامج هو توضيح الفرق بين العنصر string[i] والمؤشر string . فالأول عبارة عن قيمة عددية أو حرفية ، و الثاني هو مؤشر لبداية النضيد.

أي أن :

string[0] يكافئ *string

و

string[1] يكافئ *(string + 1)

وهكذا فإن

string[i] يكافئ *(string + i)

```
main()
{
	char string[]="Pointers are important";
	char c;
	int count=0 , i;
```

```
        printf("\n enter a letter-->");
        c=getche();
        for(i=0; i< strlen(string) ; i++)
                if( c==*(string+i) ) count++;
        printf("\n the letter %c appears in the string:",c);
        printf("\n (%s)",string);
        printf("\n %d times",count);
}
```

الشكل (8.2.3)

مثال (8.2.4) : إذا كان k و p عنوانين لموقعين في الذاكرة ، ما معنى كل من :

(a) $*(p + k)$

(b) $*p + *k$

في التعبير الأول أضفنا k إلى p ، ثم أخذنا القيمة الواقعة في العنوان (p + k) أما في التعبير k* + p* فقد أضفنا القيمة في الموقع k إلى القيمة في الموقع p ، وبالتالي فإن التعبيرين مختلفان في المدلول .

مثال (8.2.5) : ما الفرق بين (++p)* و ++p* ؟

الإجابة : لا يوجد فرق بين الاثنين . أي أن الأسبقية في التنفيذ هي للمؤثر ++ على المؤثر اللامباشر * .

مثال (8.2.6) : إذا كان p مؤشراً لبداية النضيد st فما الفرق بين الجملة

```
printf( " %s " , st)  ;
```

والجملة

```
printf( " %s " , p ) ;
```

الإجابة : لا يوجد فرق، والجملتان تؤديان نفس العمل . وإذا أردنا طباعة قيمة المؤشر، فنستخدم النضيد " %x " للنظام السادس عشري ، أو " %d " للنظام العشري على النحو :

```
printf( " %d " ,  p )  ;
```

8.3 المؤشرات والقوائم

نلاحظ أولا أن التعامل مع قائمة من الأسماء يعنى التعامل مع مصفوفة من النضائد ، وحيث أن كل نضيد هو مصفوفة من الرموز ، فإن القائمة هي عبارة عن مصفوفة ذات بعدين 2- dimensional .

فمثلا يمكننا أن نعلن أن list هي مصفوفة من الأسماء على الصورة التالية :

```
char   list [10][20] ;
```

وهى تعنى أن لدينا قائمة من 10 أسماء وطول كل اسم 20 حرفا .

ولكن يمكننا أيضاً أن نعلن list على النحو التالي :

char (∗list) [20] ;

حيث list هو عبارة عن مؤشر لمصفوفة ذات 20 عنصر ، وكذلك

(list + 1)

هو مؤشر لمصفوفة ثانية ذات 20 عنصر ، وهكذا فإن

(list + 9)

هو مؤشر لعاشر مصفوفة ذات 20 عنصرا .

مثال (8.3.1) : ما هو ناتج تنفيذ البرنامج بالشكل (8.3.1) ؟

```
#define  N 5
#define  L 12
main()
{       char (*name)[L], temp[L] ;
        int i ,j, k  , sorted;
        for(i=0; i<N ; i++)
        {
                printf("\nenter name[%d] ",i);
                scanf("%s",name+i);
        }
        for(k=0; k<N; k++)
        {       sorted=1;
```

```
                for(i=0; i<N-1 ; i++)
                {
                  if(strcmp( name+i+1 , name+i ) <0 )
                        {       sorted = 0;
                                strcpy(temp, name+i );
                                strcpy(name+i , name+i+1);
                                strcpy(name+i+1 , temp);
                        }
                }
                if(sorted) break;
        }
        printf("\n The sorted array after %d iterations",k);
        for(i=0; i<N ; i++)
                printf("\n %s",name[i]);
}
```

الشكل (8.3.1)

نلاحظ في هذا البرنامج قراءة المصفوفة name على الصورة :

```
scanf ( " %s " , name + i )  ;
```

حيث لم نستخدم هنا المؤشر & لأن name + i هي عنوان وليس متغيرا ، أي أن :

&name[i]   تكافئ   name + i

وكذلك نلاحظ كتابة name + i مرتين في جملة الطباعة

```
printf ( " \n %s  \t  \%d " , name + i  ,  name + i ) ;
```

المرة الأولى بالهيئة %s لطباعة النضيد الذي يشير له name + i
والمرة الثانية بالهيئة %d لطباعة عنوان النضيد في الذاكرة .
لاحظ استخدام \t لغرض الجدولة tab .

8.4 المصفوفة ذات البعد المتغير

مثـال (8.4.1) : اكتب برنامجا لحساب متوسط درجات عدد n من الطلبـة ، وحساب الفرق بين كل درجة والمتوسط .

سبق أن ناقشنا هذا المثال في البند (6.1) ، وبينا ضرورة استخدام المصفوفة في حل هذه المسألة. ولكن الغرض من إعادة مناقشة هذه المسألة هو التساؤل عن حجم المصفوفة ، لأن عدد الطلبة هنا غير معلوم ، وقد يكون أكبر من الحجم الذي تخصصه للمصفوفة، مما يسبب فشل البرنامج .
و لحل هذه المشكلة نستخدم الدالة malloc (اختصار memory allocation أي تخصيص الذاكرة) ، وهى معرّفة في ملف العنوان alloc.h . تقوم هذه الدالة

باستقبال عدد البايت المطلوب تخصيصها في الذاكرة ، وترجيع مؤشر لبداية هذا الحيز المطلوب.

في البرنامج (8.4.1) لدينا n درجة grade ، وكل درجة تتطلب 4 بايت، أي sizeof (float)=4 ، أي أننا نحتاج إلى حيز 4*n بايت لتخزين كل الدرجات ، أو بصورة عامة

الحيز المطلوب = n *sizeof (float)

ونظراً لأن grade هي مصفوفة عناصرها من النوع float ، فإن عملية حجز الذاكرة تتم على النحو التالي :

```
grade = ( float * ) malloc ( sizeeof( float ) *  n    ;
```

لاحظ ضرورة وجود التوجيه

```
# include   < alloc.h >
```

لتعريف الدالة malloc .

لاحظ أيضا أن n من النوع size_t ، وهو نوع معرف أيضاً في الملف alloc.h ، ويستخدم لحساب حيز من الذاكرة .

```
#include <alloc.h>
main()
{
	int i ;
	size_t n;
	float *grade, sum, ave;
	printf("\n How many students? ");
	scanf("%d",&n);
	grade=(float *)malloc(sizeof(float)*n);
	for(i=0, sum=0; i<n; i++)
	{	printf("\n enter grade %d->",i+1);
		scanf("%f",&grade[i]);
		sum += grade[i];
	}
	ave=sum/n;
	printf("\n \t grade \t grade-ave");
	for(i=0; i<n ; i++)
		printf("\n \t %5.2f \t %5.2f ", grade[i],grade[i]-
ave);
}
```

الشكل (8.4.1) برنامج مصفوفة ذات بعد متغير

8.5 تمارين

ا . ما معنى المصطلحات التالية :

operator	address-	pointer -
string pointer	operator	indirection
	null pointer	string concatenation

ب . في البرامج التالية استخدم المؤشرات كلما أمكن ذلك :

1) اكتب برنامجا يقوم بقراءة عدد صحيح ، وطباعة هذا العدد ، والموقع الذي تم تخزينه فيه .

2) اكتب برنامجا يقوم بقراءة 10 قيم من النوع double ، وطباعة متوسطها.

3) اكتب برنامجا يقوم بقراءة نضيد لا يزيد طوله عن 20 رمزا ، وتعيينه للمتغير str ، ثم طباعة الآتي :

. عدد الحروف في النضيد (الفراغات لا تحسب) .

. عدد الفراغات في النضيد .

. العدد الإجمالي للرموز في النضيد .

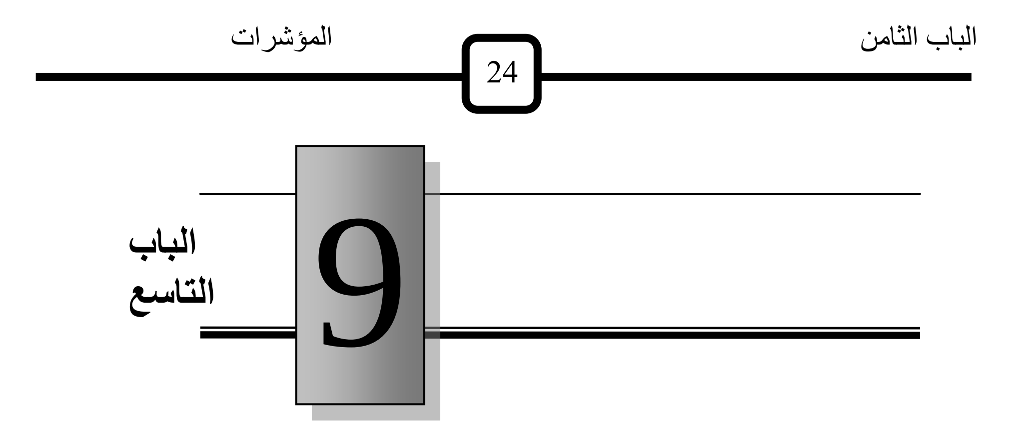
4) اكتب برنامجا يقوم بقراءة نضيد وطباعته معكوسا ، أي يبدأ من الحرف الأخير ثم الذي قبله وهكذا إلى أول حرف .

5) بدون استخدام الدالة strcmp ، اكتب برنامجا يقوم بعمل هذه الدالة ، أي قراءة نضيدين ، واختبار ما إذا كانا متكافئين أو يسبق أحدهما الآخر في الترتيب .

6) بدون استخدام الدالة strcat اكتب برنامجا يقوم بعمل هذه الدالة ، أي قراءة نضيدين وربطهما في نضيد واحد .

7) اكتب برنامجا يقوم بقراءة أسماء ودرجات عدد N من الطلبة ، وإيجاد الطالب ذي أعلى درجة (حيث N = 10) .

8) اكتب برنامجا لقراءة نص text (به n رمز) ثم طباعته ، استخدم الدالة malloc لتخصيص الحيز المطلوب لتخزين النص ، واستخدم الدالة getche لقراءة كل رمز ، والدالة putch لطباعة كل رمز .



الدوال

Functions

9.1 مقدمة

كلما تقدم المبرمج في مجال البرمجة وجد الحاجة أكثر لتنظيم برنامجه و تنسيقه، بحيث تسهل عملية إعداده وتعديله واستيعابه، فالتطبيق الواحد قد ينقسم إلى مجموعة مهام، وقد تتكرر المهمة الواحدة في اكثر من جزء من التطبيق .
من هنا جاءت الحاجة إلى مفهوم (البرنامج الجزئي)في البرمجة بصورة عامة ويطلق عليها في لغة سي (الدوال) ، فالدالة هي فرع من البرنامج العام يقوم بمهمة معينة كلما استدعي الأمر ذلك.
وبهذه الطريقة يقتصد المبرمج الكثير من التكرار في جمل البرنامج، ويصبح برنامجه أكثر قابلية للقراءة والمتابعة والتعديل .

9.2 الدالة من النوع الفارغ void

لننظر إلى الشكل (9.2.1) حيث يحتوي على برنامج رئيسي main( ) متكون من جملتين فقط هما :

```
void   welcome( ) ;
welcome( )   ;
```

من welcome عن أن الدالة declarationالجملة الأولى هي عبارة عن إعلان
(أي لاشئ) . void ، أي أن هذه الدالة ترجع لنا قيمة فارغة voidالنوع الفارغ
حيث يتم هذا welcome للدالة callأما الجملة الثانية فهي عبارة عن استدعاء
الاستدعاء على الصورة

welcome ( ) ;

ومعنى الاستدعاء هو أن يتحول المصرّف سي إلى هذه الدالة ، وينفذ ما بها من
. لننظر mainأوامر ، ثم يرجع إلى البرنامج المستدعى وهو في هذه الحالة الدالة
نفسها حيث نجد welcomeالآن إلى الدالة

```
void  welcome ( )
               {
  printf ( " \n  welcome to the function lesson " ) ;
               }
```

تتكون هذه الدالة من الاسم والنوع (بدون فاصلة منقوطة)

void welcome ( )

، وهو عبارة عن جمل الدالة محصورة body functionيلي ذلك جسم الدالة
، وفى هذا المثال لا توجد إلا جملة واحدة هي جملة طباعة { }بين القوسين
النضيد :

" welcome to the function lesson "

سينتج عنه طباعة هذا النضيد .welcomeأي أن استدعاء الدالة

ملاحظة :

الموضوعة بعد اسم الدالة ضرورية رغم أنها فارغة ولا تحتوى على ( )الأقواس شئ. سوف ندرس بعد قليل كيف نضع بين هذه الأقواس بارامترات (متغيرات) كوسيلة لتمرير بعض القيم بين الدالة المستدعية والدالة المستدعاة .

```
main()
{
        void welcome();
        welcome();
}

void welcome()
{
  printf("\n Welcome to the function lesson.");
}
```

الشكل(9.2.1) برنامج يحتوي على دالة

9.3 المتغير العام global variable

المتغير العام هو المتغير المشترك بين جميع الدوال ، أي أنه متغير يتم تعريفه في الشكل التالي nameخارج الدوال في بداية البرنامج . فمثلا المتغير الحرفي يعتبر متغيراً عاماً .

```
char  name [20]  ;
main(   )
{
 ...........
}
void   getname(   )
{
...........
}
```

أو الدالة main في الدالة nameفي هذه الحالة يمكن استخدام المتغير على حد سواء ، ويكون له نفس المدلول في الدالتين أو أي دالة أخرى getname إن وجدت بنفس الملف .

<u>مثال(9.3.1)</u> اكتب برنامجا يتكون من 3 : الدالة) إلى جانب main دوال (وهى :

1 الدالة . welcome: لطباعة النضيد

" Welcome to the function lesson "

2 . الدالة getname: لقراءة اسم لمستخدم بعد سؤاله على النحو

" What is your name ?"

3 . الدالة thanks: لطباعة النضيد

" Thank you Mr ...... for using the function lesson "

حيث يطبع اسم المستخدم في الفراغ المبين .

بين الشكل (9.3.1) البرنامج المطلوب ، حيث نلاحظ أن النضيد name هو متغير عام global تم تعريفه قبل الدالة main ، وبذلك يمكننا استخدامه في أي دالة من الدوال الثلاث ، وبالتحديد تم استخدامه في الدالتين getname والدالة thanks.

أما الدالة الرئيسية main فتحتوى على تعريف الدوال الثلاث ، وهى هنا من النوع الفارغ void ، ثم استدعاء الدالة getname لقراءة الاسم ، ثم استدعاء الدالة thanks لشكر المستخدم مع طباعة اسمه.

```
char name[20];
main()
{
        void welcome();
        void getname();
        void thanks();
        welcome();
        getname();
        thanks();
```

```
}
void welcome()
{
        printf("\n Welcome to the function lesson.");
}

void getname()
{
        printf("What is your name? ");
        gets(name);
}

void thanks()
{
        printf("\n Thank you Mr. %s for using the function
lesson.",name);
}
```

الشكل(9.3.1) برنامج يتكون من دالة رئيسية و 3 دوال أخرى

9.4 المتغير المحلى local variable

رأينا أن المتغير العام يتم إعلانه خارج الدوال ، فماذا يحدث إذا تم تعريفه داخل دالة ما؟

إذا تم إعلان متغير داخل إحدى الدوال فإنه يعتبر متغيراً محلياً خاصاً بتلك الدالة ، ولا علاقة له بالمتغير المعلن في دالة أخرى حتى لو كان يحمل نفس الاسم . يبين الشكل (9.4.1) أحد الأخطاء الشائعة في البرمجة ، وهو الخلط بين متغير محلي لدالة ما وبين متغير محلي لدالة أخرى. فمثلا لو عرفنا المتغير k في الدالة الرئيسية main على الصورة :

```
main(   )
{  int  k = 5  ;
void  fun (  )  ;
........
                    }
```

في هذه الحالة k هو متغير محلى خاص بالدالة الرئيسية ، وإذا حاولنا طباعة قيمة k في دالة أخرى مثل : فإن

```
void   fun (  )
{   int    k  ;
printf ( "\n %d " , k ) ;
}
```

رغم ( ) fun غير معروفة القيمة في الدالة k فسيعطي المصرف سي إنذارا بأن أنها قد تم تحديدها في الدالة الرئيسة .

```
main()
```

```
{
        int k=5;
        void fun();
}
void fun()
{
        int k;
        printf("\n %d",k);
}
```

مثال لخطأ شائع في استخدام المتغيرات المحلية(9.4.1)الشكل

أي أن المتغير المحلى يؤدى عملاً خاصاً بالدالة الوارد بها ولا علاقة له بالدوال الأخرى.

مثال(9.4.2) : ماذا يطبع البرنامج المبين بالشكل 9.4.2؟

. x هو main ، ومتغير محلى في الدالة yلدينا في هذا البرنامج متغير عام هو
. يجب أن نلاحظ هنا أن x , y فيوجد بها متغيران محليان هما f1أما الدالة
المتغير المحلى x لم يؤثر على قيمة كل منهما ، لأن f1 في الدالة xتغيير قيمة
فهو متغير yأما المتغير .main في الدالة x لا علاقة له بالمتغير f1في الدالة
عام لا يؤثر عليه متغير بنفس الاسم في دالة أخرى .
هو (9.4.1)وبالتالي فإن ناتج تنفيذ البرنامج

1.300000 2.500000

أي أن الاستدعاء f1 ( ) لم يؤثر على قيم x و y في الدالة main.
وبصفة عامة يجب استخدام المتغيرات العامة بحذر لأنها قد تتضارب مع متغيرات محلية بنفس الاسم في أحد دوال البرنامج .

```
float y=1.3;
main()
{       float x=2.5;
        void f1();
        f1();
        printf("\n %f %f ",x,y);
}
void f1()
{
        float x=3.4, y=5.6;
}
```

الشكل(9.4.2) المتغير المحلى

9.5 تمرير القيم إلى الدالة

نأتي الآن إلى السؤال : لماذا نضع القوسين ( ) أمام اسم الدالة ؟
والجواب : أنه عادة ما يكون للدالة بارا مترات (متغيرات) تعتمد عليها في طريقة عملها . فمثلاً الدالة الرياضية :

$$f(x) = x^2 + 3x - 2$$

على سبيل المثال فإن 4 القيمة x . فإذا أعطينا xتعتمد في قيمتها على المتغير الدالة تأخذ القيمة :

$$f(4) = 16 + 12 - 2 = 26$$

في لغة سي ، يجب تحديد نوع بارا مترات الدالة وكذلك الدالة نفسها . فمثلا التحديد

```
float   f (float  x)    ;
```

، وأنها ذات بارا متر واحد من النوع العائم float من النوع العائم fيعني أن الدالة أيضاً .

لاحظ أن الدالة يمكن أن يكون لها أكثر من متغير واحد ، ولكنها تقوم بحساب إلى الدالة returnقيمة واحدة في اسمها ، وتقوم بترجيعها عن طريق الأمر المستدعية ، كما في المثال التالي :

مثال(9.5.1)؟ ماذا يطبع البرنامج التالي :

على النحو التالي :10 إلى 1عند تنفيذ هذا البرنامج سيطبع مربعات الأعداد من

```
1      1
2      4
3      9
4     16
...   ...
10   100
```

```
main()
{
        int x,y;
        int square(int  x);
        for(x=1; x<=10; x++)
        {
                y=square(x);
                printf("\n %d  %d ",x,y);
        }
}
int square(int k)
{       int z;
        z=k*k;
        return(z);
}
```

دالة ذات متغير واحد(9.5.1)الشكل

من الأخطاء الشائعة التي تحدث عند تمرير قيمة بارا متر هو عدم توافق الأنواع ، كأن يستخدم الاستدعاء

y = square(2.5)   ;

بينما الدالة square معرّفة على أنها ذات متغير من النوع الصحيح . ويحدث هذا الخطأ بصورة خاصة عند وجود العديد من البارامترات .

لاحظ أيضاً عدم ضرورة تطابق الأسماء . فمثلاً استخدامنا في الدالة square المتغير k في الاستدعاء . وما يحدث هنا هو أن قيمة x ، بينما يقابله المتغير المتغير x ، ولهذا يسمى الاستدعاء square في الدالة k تتحول إلى المتغير

، وفيه تنتقل القيمة من الدالة call - by - value هنا الاستدعاء بالقيمة
عبر called function إلى الدالة المستدعاة calling function المستدعية
بارا متر الدالة .

مثال (9.5.2) : اكتب دالة تقوم باستقبال قيمتين من النوع float وترجع أكبرهما ، واستخدمها لحساب أكبر قيمة من بين 10. قيم موجبة

في كتابة هذا البرنامج نستخدم الخوارزمية التالية :
1 . ابدأ بأكبر قيمة y. تساوى الصفر (لأن الأعداد المدخلة لا تقل عن الصفر)
2 . اقرأ قيمة x.
3 . استخدم دالة max لحساب القيمة الأكبر من بين x و y وعيّنها للمتغير y
4 . الرجوع إلى الخطوة (2). عشر مرات

دعنا نستخدم في هذا البرنامج الدالة max: بالصورة

float max (float , float)

حيث لم نحدد أسماء المتغيرات لأن وضع الأسماء عند إعلان الدالة غير ضروري ولكن الضروري هو تحديد نوعها .

لاحظ أيضاً أن الدالة max تقوم بترجيع return قيمة واحدة وهى إما x أو y بناء على أيهما أكبر وذلك باستخدام جملة if: على الصورة

```
if ( x > y )
        return (x)   ;
else
        return (y)  ;
```

```
main()
{       float x, y=0;
        int i;
        float max(float, float);
        for(i=1; i<=10; i++)
        {
                printf("\n enter value%d-->",i);
                scanf("\n %f",&x);
                y=max(x,y);
        }
        printf("\n The maximum value is %f",y);
}
float max( float x, float y )
{
        if( x>y )
                return(x);
        else
                return(y);
}
```

الشكل(9.5.2)دالة ذات متغيرين

9.6 استخدم الماكرو Macro

(يمكننا 9.5.1 في البرنامج maxإذا كانت الدالة بسيطة التركيب (مثل الدالة
وذلك باستخدام التوجيه macroتعريفها بما يعرف بالماكرو
define

على الصورة التالية :f فمثلاً إذا عرّفنا الدالة

define f(x) 2∗ x+1

على النحو التالي :2 ∗ x + 1 بدلا من f(x)يمكننا استخدام الدالة

```
# define    f(x)     2* x+1
main(  )
{  float   x = 2.5  ,  y   ;
y = f (x)   ;
printf ( "\n  x = %f  ,  y = %f " , x , y )  ;
}
```

عند تنفيذ هذا البرنامج نحصل على النتائج

x = 2.500000 y = 6.000000

مثال(9.6.1) : ما هو ناتج تنفيذ البرنامج التالي؟ :

```
#define F(X)    5*X*X+1
main()
{  int i;
   float x;
   printf("\n\n");
   for(i=1;i<=10;i++)
   {
      x= i*0.1;
      printf("\n %5.2f %5.2f ",x, F(x) );
   }
}
```

الناتج هو :

```
0.10          1.05
0.20          1.20
0.30          1.45
 0.40          1.80
0.50          2.25
0.60          2.80
0.70          3.45
0.80          4.20
0.90          5.05
1.00          6.00
```

هناك استفادة أخرى من الماكرو (إلى جانب استخدامه بما يشبه الدالة) وهي اختصار بعض الجمل التي كثيراً ما ترد في لغة سي . فمثلاً يمكن استخدام الدالة :

```
readf (x)   ;
```

بدلا من :

```
scanf ( " %f " , &x ) ;
```

إذا قمنا بتعريف الماكرو التالي :

```
# define    readf (x)     scanf ( " %f " , &x )  ;
```

أي وإذا لم يكف سطر واحد لتعريف الماكرو يمكنك استخدام الرمز \ (backslash للاستمرارية ،كما في التوجيه التالي)

```
# define    PR (x)   \
printf ( "\n %f " , x )  ;
```

الذي يجعل الجملة :

```
PR (x)   ;
```

لها نفس مفعول الجملة :

```
printf ( " \n %f " , x )   ;
```

9.7 المصفوفة كمتغير لدالة

هل يجوز أن نبعث بمصفوفة كاملة إلى دالة ما ؟ أم أنه لا بد من أن نرسل العناصر واحداً تلو الآخر لهذه الدالة ؟
كلا الأمرين جائز في لغة سي (وفى معظم اللغات الأخرى) . فمثلاً إذا أعلنا الدالةmax: على الصورة

```
float   max ( int n   ,   float x[ ] )  ;
```

فمن الواضح هنا أن المتغير xليس متغيراً عادياً بل هو مصفوفة . وعند استدعاء هذه الدالة نستخدم اسم المصفوفة فقط (بدون أقواس) على النحو التالي مثلاً :

```
y = max ( 9 , x )  ;
```

حيث العدد 9هو قيمة n.

مثـال (9.7.1) اكتب الدالة max من النوع float التي توجد أكبر عنصر للمصفوفة xالتي تتكون من n + 1 عنصر من النوع float . واستخدم هذه الدالة لإ يجاد وطباعة أكبر عنصر لمصفوفة تتكون من 10. عناصر

يبين الشكل (9.7.1) البرنامج المطلوب ، وفى هذا البرنامج يجب أن نلاحظ الآتي :

1 . يطلب البرنامج إدخال 10 قيم على النحو التالي (كمثال) :

enter x[0] → 34
enter x[1] → 52
enter x[2] → 60
....
enter x[9] → 45

2 تم استدعاء الدالة max: بعد تكوين المصفوفة ، على النحو :

y = max (n , m) ;

حيث وضعنا اسم المصفوفة بدون الأقواس [ ] في المكان المناظر في تعريف الدالة . ولكن وضع الأقواس أمام اسم المصفوفة ضروري في تعريف الدالة حتى نبين أن هذا المتغير هو رمز لمصفوفة وليس متغيراً عاديا .

3 المتغير i والمتغير y في الدالة main متغيران محليان ، وليس لهما علاقة بالمتغيرين y و i في الدالة max. ولذلك يجب تعريفهما في الدالتين ،

```
main()
{
	float x[10], y;		int i, n=9;
	float max (int n, float x[]);
	for(i=0; i<n; i++)
	{	printf("\n enter x[%d]-->",i); scanf("%f",&x[i]);
	}
	y=max(n,x);
```

```
        printf("\n maximum = %f",y);
}
float max( int n, float x[])
{    int i;       float y=x[0];
        for(i=1; i<n; i++)   if( x[i] > y ) y=x[i];
        return(y);
}
```

الشكل (9.7.1)

9.8 تمرير قيم من الدالة

عرفنا كيف نحصل على قيمة واحدة من الدالة ، ولكن ماذا لو نريد أن نحصل منها على أكثر من قيمة ؟

مثلاً نريد من الدالة أن تحسب لنا متوسط درجات مجموعة من الطلبة وعدد الذين تحصلوا على درجة أكبر من المتوسط .

لنطلق على هذه الدالة اسم ave . لاحظ أولا أن إعلان هذه الدالة يتم على النحو التالي :

```
float   ave ( float g[  ] ,  int   n  ,  int   *kp )
```

حيث

g[ ]	مصفوفة الدرجات التي نمررها للدالة .
n	عدد الطلبة الذي نمرره أيضاً للدالة.
kp	مؤشرلعدد الطلبة الذين تحصلواعلى أكبر من المتوسط .

ملاحظة:

البارامتر الذي يرجع لنا قيمة من الدالة يجب أن يكون من النوع المؤشر pointer.

لاحظ ثانياً أن استدعاء الدالة ave: يكون على النحو التالي

```
a = ave ( g , n , &k) ;
```

حيث k هو عدد الطلبة الذين تحصلوا على درجة أكبر من المتوسط ، أي أن التمرير يكون بعنوان k وليس k. نفسه
ولهذا يسمى هذا النوع من التمرير (الاستدعاء بالعنوان call-by-address) أو (الاستدعاء بالمرجع call-by-reference) تمييزاً له عن الاستدعاء بالقيمة call-by-value . أي أن الاستدعاء بالعنوان يؤثر على القيمة الموضوعة في ذلك العنوان ، أما الاستدعاء بالقيمة فلا يؤثر على قيمة المتغير الذي تمرر قيمته (وليس عنوان) إلى الدالة.

يبين الشكل (9.8.1) مثالاً لبرنامج الاستدعاء بالعنوان وفيه نلاحظ ما يلي:

1 يتكون البرنامج من الدالتين main و ave ، حيث تتم في الدالة الأولى قراءة عدد الطلبة والمصفوفة (n) g لدرجاتهم ، وفى الدالة الثانية يتم حساب متوسطهم وفى نفس الوقت حساب عدد الطلبة k الذين تحصلوا على درجات فوق المتوسط .

k لأننا نريد تمرير قيمة ave في الدالة k للمتغير kp . تم استخدام المؤشر 2
الدالة منave إلى الدالة المستدعية main.

```
main()
{
        float g[12], a ;
        int n ,i, k;
        float ave( float g[], int n , int *kp);
        printf("\n How many students? ");
        scanf("%d",&n);
        for(i=0; i<n; i++)
        {       printf("\n enter grade[%d]-->",i);
                scanf("%f",&g[i]);
        }
        a=ave(g,n,&k);
        printf("\n average=%f",a);
        printf("\n number of students above average=%d",k);
}
float ave(float g[],int n , int *kp)
{
        float sum=0, a;
        int i;
        *kp=0;
        for(i=0; i<n ; i++)
                sum += g[i];
        a=sum/n;
        for(i=0; i<n ; i++)
                if( g[i]>a) (*kp)++;
```

```
        return(a);
}
```

برنامج الاستدعاء بالعنوان(9.8.1)الشكل

. لاحظ عدم جواز وضع (a) return هي جملة ave . آخر جملة في الدالة 3 سيتم return ، لأن أي جملة ترد بعد جملة kp*هذه الجملة قبل حساب إهمالها ولا تحسب .

والآن قد يتساءل الدارس : هل تمرير مصفوفة إلى دالة يعتبر استدعاء بالقيمة أم بالعنوان ؟

يعنى [ ] xوالسبب وراء هذا التساؤل هو أننا لاحظنا من قبل أن إعلان مصفوفة وبالتالي فإن الاستدعاء[0] x هو عنوان العنصر xأن

```
fun (x)   ;
```

فسينتج funهو استدعاء بالعنوان . وإذا تم تغيير المصفوفة المناظرة داخل الدالة تغيير في المصفوفة نفسها .

مثال(9.8.2) ماذا يطبع البرنامج المبين بالشكل : (9.8.2)؟

إن ما يطبعه هذا البرنامج هو القيم

m [0] = 55 m [1] = 66 m [2] = 88

. معنى ذلك أن تغيير main في الدالة mوليس القيم التي عينت للمصفوفة

. وهذا إثبات main أدى إلى تغييرها في الدالة funالمصفوفة في الدالة

للملاحظة التي أشرنا إليها سابقاً .

```
main()
{       int m[3]={ 12 , 23,  44} ;
        void fun( int m[] );
        fun(m);
        printf("\n m[0]=%d  m[1]=%d  m[2]=%d ",
m[0],m[1],m[2]);
}

void fun( int m[] )
{
        m[0]=55;
        m[1]=66;
        m[2]=88;
}
```

الشكل (9.8.2) برنامج استخدام المصفوفة في الاستدعاء بالعنوان

9.9 استدعاء الدالة لنفسها recursion

هل يجوز أن تستدعي الدالة نفسها ؟

نعم يجوز ذلك في لغة سي ، ويسمى هذا النوع من الاستدعاء بالتتابع recursion . وهو أسلوب في البرمجة قد يجد فيه الدارس شيئا من صعوبة الاستيعاب في البداية ، ولكن بشيء من التركيز قد يجد فيه وسيلة ممتعة للاستفادة من الحاسوب في تكرار عمل معين .

لنأخذ مثلاً الدالة :

```
void    p(int i )
{   printf ( " \n %d " ,  i )  ;
}
```

هذه الدالة تقوم بطباعة العدد الصحيح i.

ماذا لو وضعنا لها استدعاء لنفسها كالآتي :

```
void   p (int     i)
{     printf ( " \n %d " , i )  ;
p (i)  ;
}
```

لو نفذنا هذه الدالة مثلا بالاستدعاء

```
p (5)   ;
```

فإنها تقوم بطباعة العدد 5 مالا نهاية من المرات!!

ومعنى ذلك أن استدعاء الدالة لنفسها جائز في لغة سي ، ولكن يجب أخذ الحذر عند التعامل مع الدوال التتابعية recursive (أي التي تستدعي نفسها) ، فقد تدخل في حلقة لانهائية لا خروج منها .

مثال (9.9.1)

يمكن استخدام الدالة p في طباعة الأعداد من i إلى 1: تنازلياً كالآتي

```
void   p (int   i)  ;
{   if ( i = = 0 )  return    ;
else
printf ( " \n %d " ,  i  )  ;
p ( i - 1)   ;
    }
```

الآن يمكننا استدعاء هذه الدالة بالأمر :

p (20);

حيث تتعين قيمة i في البداية ، ثم تتناقص قيمة i للبارامتر 20 في كل مرة بفعل الاستدعاء :

p (i - 1)   ;

إلى أن يتحقق الشرط

i = = 1

عندها يتوقف استدعاء الدالة ويتم الرجوع إلى الدالة المستدعية . ويبين الشكل (9.9.1) لاحظ هنا أن وظيفة الدالة main لم تتعد الإعلان البرنامج متكاملاً . عن الدالة واستدعاءها .

```
main()
{
	void p(int i);
	p(20);
}


void p(int i)
{
	if(i==0)return;
	else
	printf("\n %d",i);
	p(i-1);
}
```

الشكل(9.9.1) دالة تتابعية recursive

مثال(9.9.2) اكتب برنامجاً لحساب مضروب n حيث :

$$n! = n(n - 1)(n - 2) \ldots\ldots\ldots\ldots (3)(2)(1)$$

وذلك باستخدام الأسلوب التتابعي recursive .

إذا أطلقنا اسم fac(n) على الدالة مضروب n: نلاحظ العلاقة التالية ،

$$fac(n) = n * fac(n - 1)$$

وذلك لأن :

$$n! = n * (n-1)!$$

وبالتالي يمكننا استخدام هذه العلاقة التتابعية كالآتي :

```
int   fac(int   n)
{    if (n = = 1)   return (1)   ;
else
return (n * fac(n - 1) )  ;
}
```

فإذا فرضنا أن n = 4 مثلاً ، سيكون أول ترجيع للقيمة return

$$4 * fac(3)$$

وهنا استدعاء للدالة fac بقيمة (n = 3): لنحصل على

$$4 * 3 * fac(2)$$

وهنا أيضاً استدعاء للدالة fac بقيمة n = 2: لنحصل على

$$4 * 3 * 2 * fac(1)$$

لها ، لنحصل أخيراً 1 مما يسبب توقف الاستدعاء وترجيع قيمة 1 = nالآن قيمة 4على مضروب.

```
main()
{

    int fac(int n);
    printf("\n factorial(5)=%d",fac(5));
}
int fac(int n)
{
        if(n==1)return(1);
        else
        return(n*fac(n-1));
}
```

برنامج حساب المضروب تتابعا(9.9.2)الشكل

، وعند تنفيذ هذا البرنامج نحصل على !5 برنامجا لحساب (9.9.2)يبين الشكل
:

factorial(5) = 120

ملاحظة : يمكننا حساب الدالة الأسية

$$f(x,y) = x^n$$

من العلاقة التتابعية

$$f(x,y) = x * x^{n-1}$$
$$= x * f(x , n-1)$$

بشرط أن يكون n. انظر التمارين . عددا صحيحا موجبا

9.10 الدوال الجاهزة

تتميز لغة سي بتوفير العديد من الدوال الجاهزة التي يمكن أن يستفيد منها المبرمج في العديد من المجالات . للتعرف على هذه الدوال وطريقة عملها يمكن لمستخدم توربو سي Turbo C أن يستعمل المفتاح F1 (halp) للحصول على قائمة بالدوال الجاهزة ، وذلك باختيار موضوع header files ملفات العناوين ، حيث يجد قائمة بالدوال التي تخص كل ملف من هذه الملفات . فمثلاً ، إذا أردنا الحصول على قائمة بالدوال الرياضية نتحول (بعد الضغط على F1) إلى ملفات العناوين ، ثم نختار الملف math.h لنحصل على قائمة بالدوال والثوابت الرياضية المعرفة في هذا الملف. ويبين الشكل (9.10.1) جدولاً لهذه الدوال ووظيفة كل منها ، مع بيان النطاق (أي مجال متغيراتها) ، والمدى (أي مجال الدالة نفسها) .

الدالة	الوظيفة	النطاق	المدى
abs	القيمة المطلقة لعدد صحيح	int	int
Acos	معكوس جيب التمام	double	$[0 , \pi]$

Asin	معكوس الجيب	double	(-π /2 , π/2)
Atan	معكوس الظل	double	(-π /2 , π/2)
Atan2	معكوس ظل (x,y)	(double,double)	(- π , π)
Atof	التحويل من رمز إلى عدد عائم	char	foalt
ceil	التقريب لأقرب أعلى عدد صحيح	double	double
cos	جيب التمام	double	double
cosh	جيب التمام الزائد	double	double
exp	الدالة الأسية e^x	double	double
fabs	القيمة المطلقة لعدد مضاعف أو عائم	double	double
floor	التقريب لأقرب عدد صحيح أو صفر	double	double
fmod	باقي قسمة عددين مضاعفين	(double,double)	double
labs	القيمة المطلقة لعدد طويل	long	long
log	اللوغاريثم الطبيعي	double	double
log10	اللوغاريثم العشري	double	double

pow(x,y)	حساب x أس y	(double,double)	double
pow(2)	حساب 10 أس x	double	double
sin	جيب الزاوية	double	double
sinh	الجيب الزائد	double	double
sqrt	الجذر التربيعي	double	double
tan	ظل الزاوية	double	double

الدوال الرياضية الجاهزة(9.10.1)الشكل

ملاحظات :

1 الدوال المثلثية . tan(x) و cos(x) و sin(x)

. فمثلاً لحساب degrees وليس الدرجاتradiansتستخدم التقدير الدائري

زاوية مقاسة بالدرجات ، يجب إجراء عملية التحويل x ، حيث sin(x)

$$\sin (x * 180/\pi)$$

حيثπ تقريباً يساوى ثابت مقدار 3.15196....

2 الدالة . acos(x) من y هي دالة معكوس جيب التمام ، وهى تعطى قيمة

في الفترةdoubleالنوع المضاعف ،

$$0 \le y \le \pi$$

3 الدالة . asin(x) ، هي معكوس الجيب ، وهى أيضاً من النوع المضاعف ولكنها تعطى قيمة y في الفترة

$$-\frac{\pi}{2} \leq y \leq \frac{\pi}{2}$$

وهذا المدى ينطبق أيضاً على الدالة atan.

4 الدالة : atan و الدالة atan2 . هناك نوعان من دالة معكوس الظل ، والفرق بينهما يكمن في عدد المتغيرات لكل منهما . فبينما تتطلب الدالة atan قيمة واحدة من النوع double ، تتطلب الدالة atan2 قيمتين (x,y) من هذا النوع. ونستخدم الدالة atan2 عندما نهتم بموقع الزاوية ، فمثلاً إذا كانت x سالبة وأيضا y سالبة فمعنى ذلك أن الزاوية تقع في الربع الثالث (أي أكبر من π وأصغر من 2/ π 3).

5 الدالة . pow(x,y) تمكننا من حساب x^y ، ولكن يجب أخذ الحذر من عملية مثل $(-2)^{0.5}$ لأننا بذلك نقوم بأخذ جذور عدد سالب وهى عملية ممنوعة .

6 الدالة . sqrt(x) تحسب الجذر التربيعى لأي عدد مضاعف x حيث x ≥ 0 .

يتطلب int نستخدم الدالة المناسبة للنوع. فالنوع x . عند حساب القيمة لعدد 7 ، أما النوع الطويل fbas ، والنوع العائم أو المضاعف يتطلب absالدالة long فيتطلب الدالة labs .

و هناك العديد من الدوال المهمة الأخرى إلى جانب الدوال الرياضية . والجدول (stdio.h) يبين بعض هذه الدوال المعرفة في الملف(9.10.2)

الدالة	الوظيفة
strtod	تحويل نضيد إلى عدد مضاعف
fcvt	تحويل عدد عائم إلى نضيد
atof	تحويل نضيد إلى عدد عائم
atoi	تحويل نضيد إلى عدد صحيح
atol	تحويل نضيد إلى عدد طويل
itao	تحويل عدد صحيح إلى نضيد
ltao	تحويل عدد طويل إلى نضيد
rand	تكوين رقم عشوائي

(stdlib.h) بعض دوال الملف (9.10.2)الجدول

مثال(9.10.1) اكتب برنامجا لحل المعادلة من الدرجة الثانية :

$$ax^2 + bx + c = 0$$

باستخدام القانون :

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

لاحظ إمكانية وجود جذور مركبة complex: وذلك عندما

$$b^2 - 4ac < 0$$

في هذه الحالة يتكون الجذر المركب من جزء حقيقي وجزء تخيلي كالآتي :

$x_1 = \frac{-b}{2a}$ الجزء الحقيقي للجذر الأول

$y_1 = \frac{\sqrt{4ac - b^2}}{2a}$ الجزء التخيلي للجذر الأول

$x_1 = x_2$ الجزء الحقيقي للجذر الثاني

$y_1 = y_2$ الجزء التخيلي للجذر الثاني

ويبين الشكل(9.10.1) البرنامج المطلوب .

```
/*   Solution of the quadratic equation:

           ax2 +bx + c =0
*/
#include <math.h>
main()
{
        double a,b,c,*x1p,*x2p,*y1p,*y2p,  x1, x2, y1, y2, d;
        char string[30];
        void roots(double a, double b,  double c,
                        double *x1p, double *x2p,
                        double *y1p ,double *y2p,
                        char string[] ) ;
        x1p=&x1;    x2p=&x2;
        y1p=&y1;    y2p=&y2;
        printf("\n enter a-->");     scanf("%lf",&a);
        printf("\n enter b-->");     scanf("%lf",&b);
        printf("\n enter c-->");     scanf("%lf",&c);
        roots(a,b,c,&x1,&y1,&x2,&y2, string);
        printf("\n Solution of quadratic equation. \n");
        puts(string);
        printf("\n x1=%f \t y1=%f \n x2=%f \t
                    y2=%f",*x1p,*y1p,*x2p,*y2p);
}
```

```
void roots(double a, double b, double c,
                double  *x1p, double *y1p, double *x2p,
                double *y2p,
                char string[])
{
        double d;
        d= pow(b,2)-4*a*c;
        printf("\n det=%f",d);
        if(d==0)
        {
                *x1p=-b/(2*a);
                *x2p=*x1p;
                *y1p=0;
                *y2p=0;
                strcpy(string," There are 2 equal real roots:");
                return;
        }
        if(d>0)
        {       *x1p=(-b+sqrt(d))/(2*a);
                *x2p=(-b-sqrt(d))/(2*a);
                *y1p=0;                 *y2p=0;
                strcpy(string,"There are two real roots:");
                return;
        }

        if(d<0)
        {
                *x1p=-b/(2*a);                  *x2p=*x1p;
                *y1p=sqrt(-d)/(2*a);            *y2p=-*y1p;
```

```
            strcpy(string,"There are 2 complex roots:");
            return;
      }
}
```

الشكل(9.10.1)برنامج حل معادلة الدرجة الثانية

(9.10.1)ملاحظات عن البرنامج

1. الدالة roots هي التي تقوم بعملية إيجاد الجذرين ، حيث تستقبل هذه الدالة المعاملاتa,b,c من النوع double: وترجع الآتي

*x1p = الجزء الحقيقي للجدر الأول

*x2p = الجزء الحقيقي للجذر الثاني

*y1p = الجزء التخيلي للجذر الأول

*y2p = الجزء التخيلي للجذر الثاني

حيث نلاحظ استخدام المؤشرات لهذه المتغيرات الأربعة لأنها تمرر قيماً من الدالةroots. إلى الدالة المستدعية

2. استخدام التوجيه .

include < math.h >

لإيجاد الجذر التربيعي للمحدد ( )sqrtوذلك نظراً لاستخدام الدالة الرياضية وكذلك الدالة،pow(b,2) لحساب b^2.

، وهو عبارة عن نضيد ( )roots إلى بارا مترات الدالة string . أضفنا 3
يكافئ الآتي :

أ . في حالة وجود جذرين حقيقيين متساويين (أي عندما

) فإن :$d = b^2 - 4ac = 0$

string = " There are 2 equal real roots "

) فإن $d > 0$ب . في حالة وجود جذرين حقيقيين غير متساويين (أي عندما

:

string = " There are 2 real roots "

) فإن :$d < 0$جـ . في حالة وجود جذرين مركبين (أي عندما

string = " There are 2 complex roots "

هو عبارة عن مصفوفة من الرموز ، وأن :stringلاحظ أن النضيد

string = & string[0]

يعتبر مؤشراً ويمكن استخدامه في الاستدعاء بالعنوان. stringوبالتالي فإن

لتعيين نضيد ، " = "كما لاحظنا سابقا لا يجوز في لغة سي استخدام المؤثر
كما في البرنامج .strcpyولكن (بدلاً من ذلك) نستخدم الدالة

مثال(9.10.2) : اكتب برنامجا يقوم بقراءة عدد من النوع الصحيح ولكن
يفترض أولاً أنه نضيد (مصفوفة من الرموز) ، ثم بعد التأكد من خلوه من أى رمز
) يقوم بتحويله إلى عدد صحيح .9 إلى 0(غير الأرقام من

البرنامج المطلوب في هذا المثال ، حيث نجد المتغيرات (9.10.2)يبين الشكل
والدوال التالية :
itemp. نضيد مؤقت لقراءة العدد المدخل :
inum عدد صحيح يكافئ : itemp. بعد عملية التحويل
0 : دالة لاختبار نضيد والتأكد من خلوه من أي رمز عدا الأرقام من check
في 0 بعد التأكد من هذه العملية وترجيع 1 . تقوم بترجيع 9إلى
الحالة الأخرى .
C متغير صحيح توضع فيه القيمة المرجعة من الدوال : check.

لاحظ أن عملية التحويل من نضيد إلى عدد صحيح

inum = atoi(itemp) ;

من الرموز itemp ، أي بعد التأكد من خلو (c = = 1)لا تتم إلا إذا كانت
الأخرى غير الأرقام .

لكـي تختبر هذا البرنامج ، أدخل أرقاماً سليمة ، لتحصل على الرسالة :

you have entered the number ....

أو أدخل أرقاما تحتوى على رمز غير رقمية لتحصل على الرسالة :

data entery error

```
#include <ctype.h>
main()
{
	char  itemp[5];
	int inum, c;
	printf("\n enter a number-->");
	gets(itemp);
	c=check(itemp);
	if(c==1)
	{	inum=atoi(itemp);
		printf("\n you have entered the number
%d",inum);
	}
	else
		printf("\n data entry error");
}
int check(char str[] )
{
	int i;
	for(i=0;  str[i] != '\0' ; ++i)
		if( ! isdigit(str[i])  ) return(0);
```

```
        return(1);
}
```

الشكل(9.10.2) برنامج يستخدم الدالة atoi

9.11 تمارين

1 . اكتب الدالة greet( ): التي تطبع النضيد

" Good morning. How are you ? "

واستخدمها في الدالة main( ).

2: ما معنى المصطلحات التالية؟ .

global variable
local variable
function

3 . اكتب الدالة getage من النوع float التي تقوم بسؤال المستخدم على النحو :

How old are you ?

ثم تقوم بقراءة عمره .

(وذلك لغرض طباعة كلمة ( )main استدع هذه الدالة في الدالة الرئيسية too (وإلا فتتم طباعة العبارة 18 إذا كان العمر أكبر من (ok young).

في هذا البرنامج ، واستخدم دالة لطباعة age ملاحظة : استخدم متغيرا عاما المخرجات .

4: ماذا يطبع البرنامج التالي؟ .

```
main(  )
{  int  k = 5
void   f( int    k )  ;
f(k)  ;
printf( " \n %d " ,  k )  ;
}
void   f( int   k )
k  =  6  ;                           {
return   ;
}
```

5: ماذا يطبع البرنامج التالي؟ .

```
float   x = 8.5
main(   )
{    void   fun( void )  ;
```

```
fun(    )
printf( " \n %f " ,  x )   ;
}
void   fun( void )
{    x = 9.5  ;
return    ;
}
```

6 . اكتب الدالة

float   tax(float   income)

التي تقوم باستقبال قيمة الدخل income وحساب الضريبة tax على النحو التالي :

الضريبة = 15 % من الدخل إذا قل الدخل عن 500.

= 20 % من الدخل إذا كان الدخل 500. فما فوق

استخدم هذه الدالة لحساب الضريبة على الدخل لعدد 10. موظفين

7 . إذا كان قبول الطالب في قسم الحاسب الآلي يعتمد على متوسطه العام ودرجته في المقرر CS111 ، بحيث لا يقل المتوسط العام عن 65 ، ولا تقل درجته في المقرر CS111 عن 50 ، اكتب دالة ترجع قيمة 1 إذا توفر الشرط ، وقيمة 0. إذا لم يتوفر أحدهما أو كلاهما

استخدم هذه الدالة في إعداد قائمة بالطلبة المقبولين من بين مجموعة المتقدمين .

8 امتحانات ، بحيث تساوى 3 . إذا كانت أعمال الفصل تحسب من درجات درجة أعمال الفصل مجموع أكبر درجتين من الدرجات الثلاث ، اكتب دالة تقوم بهذا العمل ، واستخدمها لحساب درجات أعمال الفصل لعدد من الطلبة .

9. أحسب

$$f(x) = x^2 + 2x + 3$$

أ . باستخدام الماكرو.

ب . باستخدام الدالة .

وذلك لجميع قيم x من 0 إلى 2 بزيادة ثابتة مقدارها 0.1.

10 . استخدم الماكرو لتعريف عنوان متغير في الذاكرة على النحو ADDRESS(x) ، وكذلك لتعريف &x كبديل للمؤشر المتعارف عليه AND بدلا من المؤشر && ، و OR بدلاً من ||.

واستخدم هذه التعريفات في دالة تقوم بقراءة العمر ، وتطبع كلمة " OK " إذا كان العمر أقل من 65 وأكبر من 18 ، وتطبع كلمة " ERROR " إذا كان العمر أقل من 18 وأكبر من 65.

11 اكتب الدالة sum التي تستقبل مصفوفة من النوع float وعدد عناصرها n ، وتقوم بإيجاد مجموع كل العناصر . استخدم هذه الدالة لحساب مجموع عناصر مصفوفة x عدد عناصرها 10 ، وكذلك حساب مجموع مربعات هذه العناصر .

12 استخدم الاستدعاء بالعنوان في كتابة دالة calc تقوم بقراءة قيمة المشتريات ، وحساب الربح (15% من رأس المال) وقيمة المبيعات اللازمة لتحقيق هذا الربح . استخدم هذه الدالة لإعداد قائمة أسعار الشراء والبيع بها 10 أصناف من البضاعة .

13 اكتب الدالة range التي تستقبل المصفوفة grades من النوع العائم float ، وتقوم بحساب أعلى قيمة max و أدني قيمة min والفرق بينهما . استخدم هذه الدالة في حساب أعلى وأدنى درجة ، والفرق بينهما لعدد 20 طالباً .

14 اكتب الدالة star التي تقوم بطباعة الرمز * بعدد n من المرات وذلك
أ . استخدم الأسلوب العادي (التكراري) .
ب . استخدم الأسلوب التتابعي recursive.

15 اكتـب الدالــة p(x,n) . التــي تقـوم بحســاب x^n بالأسـلوب التتــابعي . استخدم هذه n من النوع الصحيح n ، و float النوع العائم recursive الدالة في حساب $(3.5)^3$.

16 اكتب الدالة reverse . التي تقوم باستقبال string نضيد ، وتقوم بطباعته معكوساً وذلك باستخدام الأسلوب التتابعي . مثال : النضيد " book " تطبعه الدالة على النحو " koob ".

17: اكتب الدالة الجاهزة التي تقوم بحساب الآتي .

أ(-5) . القيمة المطلقة للعدد
ب(- 6.4) . القيمة المطلقة للعدد
جـ . التحويل من 6.4 إلى 7
د . التحويل من 6.7 إلى 6
هـ 26.7 . اللوغاريتم العشري للعدد
و 34.5 . اللوغاريتم الطبيعي للعدد
ز 30 درجة° . جيب الزاوية
ح $e^{2.1}$.
ط $(1.3)^{3.4}$.
ي (7.8) . الجدر التربيعي للعدد
ك . تحويل نضيد إلى عدد عائم .

18: اكتب دالة لإجراء اللعبة التالية .

يكوّن الحاسوب رقما عشوائيا بين الصفر والمائة ، ويطلب من اللاعب أن يعرف ما هو هذا الرقم ، فإذا كانت إجابته أقل من الرقم المطلوب ، تظهر على الشاشة ، وإذا كانت إجابته أكبر من المطلوب ، (higher)كلمة وهكذا حتى يحصل اللاعب على الإجابة الصحيحة (lower)تظهر كلمة .

19 لتحويل نضيد إلى عدد عائم ، وذلك بعد التأكد atof . استخدم الدالة الجاهزة من خلو العدد من أي رمز غير الأرقام والفاصلة العشرية .

20 ، float من النوع array التي تقوم باستقبال المصفوفة sort . اكتب الدالة ويتم ترجيعها مرتبة تنازليا .nوعدد عناصرها
اختبر هذه الدالة في برنامج كامل .

21 من النوع names التي تقوم باستقبال المصفوفة sortnames . اكتب الدالة) ، وتقوم الدالة بترتيب هذه nالنضيد (أي مجموعة من الأسماء عددها المصفوفة تصاعديا . اختبر هذه الدالة بمجموعة من الأسماء .